This paper describes the design concept of Responsive MultiThreaded Processor (RMTP) for distributed real-time control that controls various embedded systems, including robots, home automation, factory automation, etc. RMTP processor integrates an 8-way multithreaded processor (RMTP processing unit) for real-time processing, four sets of Responsive Link II for real-time communication, and I/O peripherals including DDR SDRAM 1/8s, DMAC, PC3164, USB2.0, IEEE1394, PWM generators, pulse counters, etc., into an ASIC chip. System designers can use various on-chip functions easily by connecting required I/Os to this chip directly. The designers can also realize distributed control systems by connecting several RMTP processors with their own functions via Responsive Link II.

Keywords: real-time, distributed controller, Responsive Link, SoC, RMT

1. Introduction

Parallel/distributed processing is required to control high-performance and/or multifunctional robots in real-time, especially when the design of a large-scale system is made up of several smaller, parallel/distributed processors or when we design a control system in which many sensors and actuators are widely distributed.

However, the distributed control systems have not been so popular, because control was thought unsuitable for parallel/distributed processing.

This paper describes the design concept of Responsive MultiThreaded Processor (RMTP) that is a system LSI to realize distributed real-time systems easily. In other words, RMTP Processor is a system-on-a-chip that integrates almost all functions required to realize distributed real-time systems in one chip.

We design and implement new mechanisms in each functional block. RMTP Processor is a very large-scale system LSI (over 10Mgate). This paper describes the design policy and overall design of the chip, since this paper cannot detail all RMTP processor design.

2. Real-Time

We designed all functional units of RMTP Processor so as to enable real-time processing/communication. Enough silicon area is used to realize real-time capability.

Here, real-time means that the exactness of the processing, communication, and operation of the system depends not only on the result but also on the time it takes to get the result, and that deadlines must be met in the narrow sense.

2.1. Hard and Soft Real-Time

Real-time is classified into hard and soft real-time. Hard real-time strictly requires a deadline to be met, and if it is not, the value of the processing suddenly becomes zero. On the other hand, soft real-time allows a slight deadline miss, and if the deadline is not met, the value decreases rapidly as time passes.

Distributed control is an example of hard real-time systems. Some sensors and actuators are connected to the system via networks, so if a sensor node cannot communicate with an actuator node in real-time and control cannot be executed within the cycle time limitation, e.g., 1ms, the actuator cannot be controlled normally and may undergo “ripple,” which, in the worst case, may cause the actuator to crash.

MPEG decoding via networks, e.g., VoD, is an example of soft real-time systems. If communication and processing for decoding cannot be done within the cycle time limitation, e.g., 33ms, we cannot watch moving picture normally, i.e., block noise may occur, picture and voice may not be synchronized, and motion may be unnatural. However, the deadline miss does not usually yield a catastrophic result.

Real-time systems are classified by application (task) as follows:

Hard real-time: Control systems require hard real-time as follows:

- Communication: Data volume is small, but latency requirements are high, i.e., latency is more important than throughput.
- Operation: Calculation volume is small, but time...
limitations must be met.

Time Tick Resolution and Deadline: The time tick resolution and deadlines are shorter (100us−10ms) in many cases.

Soft real-time: Multimedia systems require soft real-time as follows:

Communication: Data volume, e.g., streaming, is large, but the latency requirement is not so severe. Throughput is more important than latency.

Operation: Calculation volume is relatively large, e.g., MPEG decoding, but time limitations are not so severe compared to control systems.

Time Tick Resolution and Deadline: The time tick resolution and deadlines are longer (10ms−1s) in many cases.

We design and implement RMT Processor to realize real-time systems requiring hard and soft real-time simultaneously.

2.2. Real-Time Scheduling

Real-time scheduling is required to realize distributed control because all possible cases occurring in complex systems cannot be anticipated. Conventional real-time schedulers include EDF (Earliest Deadline First) and RM (Rate Monotonic) [1]. Most real-time operating systems based on such schedulers preempt and execute tasks based on priority at every clock tick. Fig.1 shows sample scheduling based on EDF, whose scheduling policy holds that the earlier the deadline, the higher the priority.

In operation, preemption means context switching. In communication, preemption means packet overwriting. Our goal is thus to realize real-time processing/communication architecture that optimally conducts context switching and packet overwriting. Our RMT Processor executes operations with suitable priority and sends/receives packets with suitable priority given by real-time schedulers.

3. Design Policy

Real-time schedulers translate deadlines or cycle time to priority levels. The design policy of RMT Processor holds that each functional unit has overtaking/arbitrating capability based on priority. To realize this policy, sufficient silicon area is used. In order to realize real-time functions, some functional units use large silicon area needed for implementing their real-time capabilities as much as needed to realize their primary functions.

Cache, instruction fetch units, instruction issue units, execution units, etc., for example, are designed to overtake/arbitrate based on priority in RMT PU. Other I/O functional units including Responsive Link II, etc., are also designed this way.

We design and tune the architecture of RMT Processor to translate the architecture into actual silicon. For example, multiple port memory can be easily designed by HDL, and it is easy to design a processor using multiple port memory. Multiple port memory needs very large silicon area, however, and signal latency become longer.

It requires much labor to design multiple port memory itself, so we restrict design, using dual port memory as multiple port memory to realize realistic design. When we design each functional unit, we restrict design to realize a realistic VLSI chip.

Back-end design including layout and P&R (Place and Route) is important in designing the chip using the latest VLSI process rule. Whether the chip actually runs and whether it operates properly depend on the co-design of front-end and back-end design, so we carefully co-design the architecture of RMT Processor.

Since many embedded systems are battery-driven, low power consumption is an issue. Current distributed control systems require soft real-time processing including moving image and voice processing requiring high processing power.

Such high-performance processing consumes much power, so we design RMT Processor to balance these requirements.

We design RMT Processor based on the following policy:

Sufficient silicon area is used to realize real-time capability by hardware. Every functional unit has overtaking/arbitrating mechanism based on priority. Realistic architecture design (back-end and front-end) is sought. We try to balance high-performance/multi-functional processing and low power consumption.

4. RMT Processor Design

4.1. Overview

Responsive Processor, a predecessor of RMT Processor, has real-time communication but no real-time processing by hardware. RMT Processor has both
real-time communication and real-time processing. RMT Processor integrates the following functions into one chip (system-on-a-chip) so that RMT Processor can be widely used in various embedded systems. Fig. 2 shows the block diagram of RMT Processor.

- Real-time processing functions: RMT PU
- Real-time communication functions: Responsive Link II
- Computer peripherals: PCI64, USB2.0, IEEE1394, UART, DDR SDRAM I/Fs, DMACs, etc.
- Control peripherals: PWM generators, pulse counters, etc.

System designers can use various on-chip functions easily by connecting required I/Os to this chip. The designers can realize distributed control systems by connecting several RMT Processors with their own functions via Responsive Link II.

While an internal vector unit processes moving images captured by a digital camera connected to IEEE1394 in real-time, pulse counters and PWM generators are controlled by the processing result and corresponding actuators are controlled.

4.2. Responsive Link

RMT Processor integrates Responsive Link II as real-time communication. Responsive Link II is based on the real-time communication standard Responsive Link [2]. The real-time communication architecture of Responsive Link II is designed for distributed real-time systems as follows:

- Separation of data transmission for soft real-time and event transmission for hard real-time
- Fixed packet size: 64-byte data and 16-byte events
- Point-to-point serial link
- Independent routing of data and event links
- Priority-based packet overtaking: The packet with higher priority overtake packets with lower priority at each node.
- Packet acceleration/deceleration using priority replacement: Packet priority can be replaced with a new priority level at each node to accelerate/decelerate packets under distributed control.
- Prioritized routing: When multiple packets with different priority are sent to the same destination, different routes can be set to realize exclusive communication links or detours.
- Variable link speed: 800, 400, 200, 100, 50, 25, or 12.5 Mb/s
- Plug & Play
- Topology-free
- Low latency

Responsive Link II realizes flexible real-time communication through these unique features. Maximum unidirectional communication speed is 800 Mb/s. Communication speed can be changed dynamically to reduce power consumption. Five Responsive Link IIs are implemented in RMT Processor. One set is connected to RMT PU directly, and the other four sets are arranged on the chip. Since Responsive Link IIs consist of an event link and a data link, each link is full-duplex; and there are five Responsive Link IIs on the chip, the total switching speed of Responsive Link II per chip is 160 Gb/s.

Responsive Link II is connected to RMT PU two ways:

One way is that Responsive Link II is treated as a normal 32-bit I/O peripheral. RMT PU reads/writes to registers of Responsive Link II. Using a DMAC and dual port memory in Responsive Link II, packets are automatically transmitted and received by hardware.

The other way is that event links required to be low latency are connected to the internal processor bus of the RMT PU directly. Since event links of Responsive Link II are treated as internal register files, RMT PU transmits and receives event packets by writing and reading from internal register files, enabling very low latency communication and inter-processor shared registers easily.

Responsive Link has been standardized as the IPSJ Trial Standard (IPSJ-TS 0006:2003), which is a domestic standard in Japan [3]. Responsive Link will be standardized by ISO/IEC JTC1 SC25. Once international standardization is realized, real-time communication among different systems is realized. Large-scale distributed control is also realized.

4.3. RMT PU

Real-time processing architecture of RMT PU, RMT Architecture, supports various real-time processing mechanisms by hardware. RMT PU has a simultaneous multithreading mechanism for real-time processing. RMT architecture is similar to SMT [4,5] with priority. The instruction set of RMT PU is upwardly compatible with MIPS. Main features of RMT PU are as follows:

- Simultaneous multithreading (SMT) with priority
Table 1. Peak performance of RMT processing unit,

<table>
<thead>
<tr>
<th>Data Type</th>
<th>1.2GIPS</th>
<th>600MFLOPS</th>
<th>9.6GIPS</th>
<th>4.8GFLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>32bit scalar integer</td>
<td>32bit vector integer</td>
<td>64bit scalar floating point</td>
<td>64bit vector floating point</td>
<td></td>
</tr>
<tr>
<td>Clock frequency</td>
<td>Max. 8W</td>
<td>Mx. 300Mhz</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 3. Bus architecture of RMT Processor.

- Simultaneous eight-thread execution
- Context cache: 32 threads
- Context switching by hardware
- 256-level priority
- Control of all functional units by priority
- Control of threads by hardware in interrupts
- Shared registers for multiple threads
- Synchronization for multiple threads

- High-performance vector processors: 2-INT, 2-FP
- Multiple threads sharing a vector unit
- Flexible compound operations defined by software

Flexible real-time processing can be achieved by these unique features.

RMT PU can execute simultaneously with performance shown in Table 1 because of the multi-threaded mechanism.

4.4. Bus

Many I/O peripherals integrated in RMT Processor are connected to internal busses as shown in Fig. 3. The wider bus width between RMT PU and memory improves performance, but the bus width of many I/O peripherals is less than or equal to 32 bits, so we classify memory and I/O peripherals as 256-bit or 32-bit bus based on their required bus width. They are connected by bus sizing. We also design DMA controllers with data sizing to realize flexible data transfer.

4.5. Power Management

RMT Processor is required to realize both high performance/real-time processing/communication and low power consumption. Power consumption is calculated as follows:

\[ P = \alpha \times f \times C_{\text{load}} \times V_{\text{dd}}^2 \]

Since power consumption is proportional to the power source voltage \( V_{\text{dd}} \), very low voltage processing \((V_{\text{dd}} = 1.0V)\) is used to reduce total chip power consumption.

We designed a power management unit that controls the clock frequency. The power management unit dynamically changes the clock frequency (256 levels), stops the clock, and resets the unit at each functional unit. Software including an operating system controls the trade-off between speed and power using this mechanism. The unit of clock distribution is determined by I/O peripherals and large functional units in RMT PU.

Power management software, for example, statically stops clocks of I/O peripherals not used by current applications to reduce power consumption. When current running threads only execute scalar operations, clocks of vector units whose power consumption is very large are stopped dynamically by power management software. When vector processing is needed, clocks are supplied dynamically.

Maximum speed is not needed to process a task in real-time. In other words, sufficient minimum speed is required to process the task in real-time. When a task decodes MPEG, for example, software including an operating system controls the power management unit to

Journal of Robotics and Mechatronics Vol.16 No.2, 2004
Table 2. Context switch.

<table>
<thead>
<tr>
<th>Software</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>59clock (save:277 + restore:313)</td>
<td>4clock</td>
</tr>
</tbody>
</table>

reduce the clock with enough computational power to reduce power consumption. As noted above, a real-time operating system can control the power management until to realize real-time processing and low power consumption.

4.6. Device Fabrication

We designed RMT Processor from front-end design to back-end design. We designed and verified the mask pattern (GDSII) and TSMC fabricated the actual chip.

- Fabrication masker: TSMC
- Process: 0.13um CMOS 8-layered Cu wiring
- Number of gates: 14Mgates
- Voltage: 
  - Core: 1.0V
  - I/O: 2.5V
- Die size: 10.0mm × 10.0mm
- Chip size: 4.0cm × 4.0cm BGA

Figure 4 shows the layout of RMT Processor, Responsive Link II is at the right upper corner. Responsive Link II uses 600Kgates. RMT PU uses 5Mgates and is at the center of the chip.

5. Evaluation

Since we cannot describe all evaluations of various functions in RMT Processor, we evaluated the performance of real-time processing in RMT PU.

5.1. Context Switch

We evaluate the overhead of a context switch. In a software context switch, the software (OS) saves and restores a context set. In a hardware context switch, hardware switches the context set between register files and context cache by the thread swap instruction (SWAPTH instruction) of RMT PU. Table 2 shows context switching costs.

RMT Processor provides very fast context switching by hardware. New operating systems unrestricted by the frequency of context switching and whose clock tick is very short will thus be developed.

5.2. Real-Time Processing Performance

We evaluate the performance of real-time processing using a quick sort as a test program. Multiple quick sort programs run on RMT Processor simultaneously and their target data is the same. They start simultaneously in parallel, so the same cache block is simultaneously accessed by multiple programs, i.e., the worst case. Fig.5 shows execution time without priority and Fig.6 shows execution time with priority. With priority, Thread 0 has the highest priority, Thread 1 has second priority, and Thread 7 has lowest priority.

In case of without-priority, as the number of threads increases, thread execution time increases because resource conflicts occurs as the number of threads increases. However, total throughput is improved.

In case of with-priority, the execution time of Thread 0 with the highest priority is constant at 40us. As the number of threads increases, no change occurs in the execution time of Thread 0. Similarly, the execution time of threads with other priority is less than constant based on the given priority. The execution time of Thread 2, for example, is less than 610us. A real-time scheduler assigns lower priority to a thread because its deadline is longer, so the lower priority thread must wait for higher priority threads to finish, even if the lower priority thread is ready to run. Thread 7 with the lowest priority is kept waiting on its register files until higher priority threads have finished execution. After Threads 0, 1, 2 have finished execution, Thread 7 starts to run simultaneously with other threads including Threads 3, 4, 5, and 6. These mechanisms for real-time execution are realized by hardware.

As noted above, RMT Processor guarantees the execution time based on priority.

Journal of Robotics and Mechatronics Vol.16 No.2, 2004

Fig. 5. Execution time without priority.

Fig. 6. Execution time with priority.
6. Conclusions

This paper described the design concept of RMT Processor for distributed real-time control. RMT Processor integrates real-time processing functions (RMT PU, Caches, etc.), a real-time communication function (Responsive Link II), computer peripherals (DDR SDRAM I/Fs, DMAC, PCI64, USB2.0, IEEE1394, etc.), and control peripherals (PWM generators, pulse counters, etc.) into a VLSI chip. RMT Processor guarantees the execution time of tasks with priority given by a real-time scheduler.

RMT Processor also guarantees the communication time of packets with priority given by a real-time scheduler. The time tick resolution is shortened. Large-scale distributed real-time systems including robots are realized by RMT processor.

Since RMT Processor is very small and easy to connect, it is easily embedded where it serves as the controller of office automation, home automation, factory automation, intelligent buildings, etc. We hope that RMT Processor and Responsive Link will be used widely in many systems.

Acknowledgements

This study was conducted through Special Coordination Funds of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References:


